Estruturas de Dados - ESP412

Profª Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal de Ciência e Tecnologia da Bahia
Campus de Feira de Santana

carolsoko@ifba.edu.br

December 5, 2024

Listas, Filas e Pilhas

Listas

Listas

Em ciência da computação, uma lista ou sequência é uma estrutura de dados abstrata que implementa uma coleção ordenada de valores, onde o mesmo valor pode ocorrer mais de uma vez.

Listas

Em ciência da computação, uma lista ou sequência é uma estrutura de dados abstrata que implementa uma coleção ordenada de valores, onde o mesmo valor pode ocorrer mais de uma vez.

Uma instância de uma lista é uma representação computacional do conceito matemático de uma sequência finita, que é, uma tupla. Uma lista é armazenada dinamicamente em memória RAM.

Listas

Uma lista possui algumas propriedades:

- Os nodos são criados dinamicamente, à medida que for necessário. Assim, a quantidade total de memória usada para a lista depende da quantidade de dados nela armazenados (compare isso com um vetor ou matriz).

Listas

Uma lista possui algumas propriedades:

- Os nodos são criados dinamicamente, à medida que for necessário. Assim, a quantidade total de memória usada para a lista depende da quantidade de dados nela armazenados (compare isso com um vetor ou matriz).
- A lista não precisa ocupar uma área de memória contígua: como nodos são alocados dinamicamente, eles podem ocupar áreas de memória arbitrárias, e não há nenhuma relação entre a localização dos nodos em memória e sua ordem na lista (novamente compare isso com um vetor ou matriz).

Listas

Uma lista possui algumas propriedades:

- Não é possível indexar os nodos, por isso para acessar um nodo deve-se obrigatoriamente procurá-lo a partir do início da lista, seguindo cada nodo até chegar àquele procurado.

Listas

Uma lista possui algumas propriedades:

- Não é possível indexar os nodos, por isso para acessar um nodo deve-se obrigatoriamente procurá-lo a partir do início da lista, seguindo cada nodo até chegar àquele procurado.
- Para adicionar um nodo, basta modificar a referência do nodo que o antecede na lista. Assim, não é necessário "empurrar" os nodos seguintes para frente (como seria o caso de um vetor).

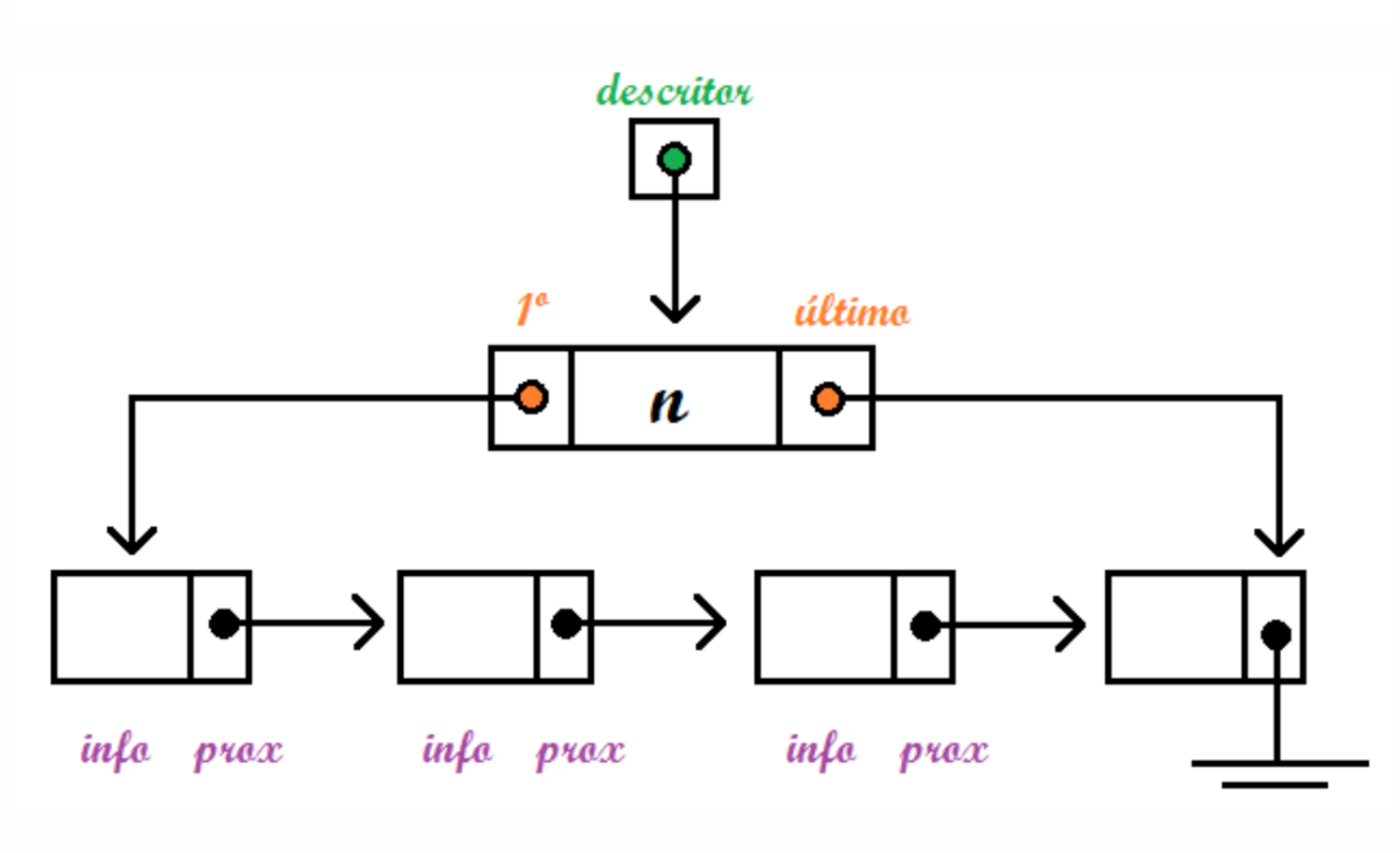
Listas

Uma lista possui algumas propriedades:

- Não é possível indexar os nodos, por isso para acessar um nodo deve-se obrigatoriamente procurá-lo a partir do início da lista, seguindo cada nodo até chegar àquele procurado.
- Para adicionar um nodo, basta modificar a referência do nodo que o antecede na lista. Assim, não é necessário "empurrar" os nodos seguintes para frente (como seria o caso de um vetor).
- Para remover um nodo é a mesma coisa: basta modificar a referência de seu nodo antecessor. Assim, não é necessário "deslocar pra trás" os nodos seguintes (como seria o caso de um vetor).

Listas

A lista possui um descritor que indica o nó inicial e o nó final da lista, além de indicar quantos elementos (n) contém a lista.



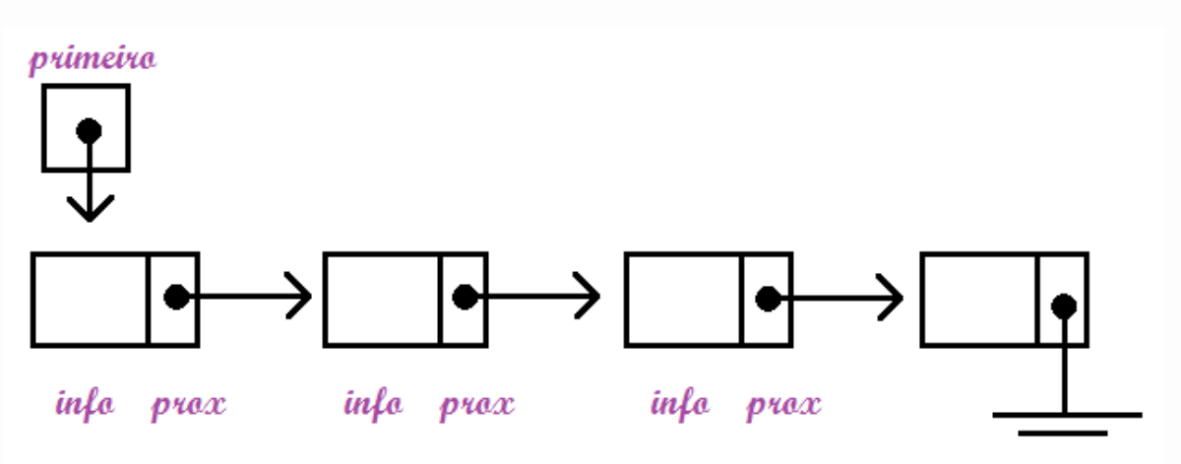
Listas Simplesmente Encadeadas

Listas Simplesmente Encadeadas

Uma Lista Simplesmente Encadeada é uma estrutura de dados em que os objetos estão organizados de forma linear, porém, diferente de um vetor, não é uma estrutura de dados indexada, é uma estrutura de dados em que os elementos são interligados por ponteiros. Desta forma, os nodos são encadeados de forma unidirecional, pode-se percorrer a lista do primeiro nodo em direção ao último nodo, mas não na direção contrária. [Cormen et al. 2009]

Listas Simplesmente Encadeadas

Uma Lista Simplesmente Encadeada é uma estrutura de dados em que os objetos estão organizados de forma linear, porém, diferente de um vetor, não é uma estrutura de dados indexada, é uma estrutura de dados em que os elementos são interligados por ponteiros. Desta forma, os nodos são encadeados de forma unidirecional, pode-se percorrer a lista do primeiro nodo em direção ao último nodo, mas não na direção contrária. [Cormen et al. 2009]



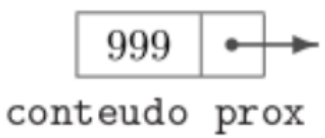
Listas Simplesmente Encadeadas

Cada nodo, célula ou elemento, da Lista Simplesmente Encadeada é composto pelo conteúdo do nodo da lista e por um ponteiro para o próximo nodo. Assim, uma Lista Simplesmente Encadeada é na verdade uma lista de registros. Em linguagem C, uma lista de *Structs*.

Listas Simplesmente Encadeadas

Cada nodo, célula ou elemento, da Lista Simplesmente Encadeada é composto pelo conteúdo do nodo da lista e por um ponteiro para o próximo nodo. Assim, uma Lista Simplesmente Encadeada é na verdade uma lista de registros. Em linguagem C, uma lista de *Structs*.

```
typedef struct reg {
    int          conteudo;
    struct reg *prox;
} celula;
```



Listas Simplesmente Encadeadas

Cada nodo, célula ou elemento, da Lista Simplesmente Encadeada é composto pelo conteúdo do nodo da lista e por um ponteiro para o próximo nodo. Assim, uma Lista Simplesmente Encadeada é na verdade uma lista de registros. Em linguagem C, uma lista de *Structs*.

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct reg{
    int         conteudo;
    struct      reg *prox;
}celula;
```

Listas Simplesmente Encadeadas

Inserir:

O interessante de estruturas de dados que utilizam com ponteiros é que estas não têm limites de tamanho. O limite é o tamanho da memória disponível para armazená-las. Cria-se o primeiro elemento, o topo, ou cabeça, e a partir deste, cresce a lista de acordo com a necessidade do usuário.

Listas Simplesmente Encadeadas

Inserir:

O interessante de estruturas de dados que utilizam com ponteiros é que estas não têm limites de tamanho. O limite é o tamanho da memória disponível para armazená-las. Cria-se o primeiro elemento, o topo, ou cabeça, e a partir deste, cresce a lista de acordo com a necessidade do usuário.

```
celula *inserir(int x, celula *p){
    celula *nova;
     nova = malloc (sizeof (celula));
     nova->conteudo = x;
     nova->prox = NULL;
    if (p != NULL)
          nova->prox = p;
    p = nova;
    return p;
}
```

Listas Simplesmente Encadeadas

Imprimir:

Exibir os elementos inseridos na lista é uma tarefa bem simples e pode ser feita tanto recursivamente como de forma iterativa. Vejamos as duas soluções possíveis:

Listas Simplesmente Encadeadas

Imprimir:

Exibir os elementos inseridos na lista é uma tarefa bem simples e pode ser feita tanto recursivamente como de forma iterativa. Vejamos as duas soluções possíveis:

```
void imprimi_r(celula *listaEncadeada) {
    if (listaEncadeada != NULL) {
        printf ("%d\n", listaEncadeada->conteudo);
        imprimi_r(listaEncadeada->prox);
    }
}
void imprimir(celula *listaEncadeada) {
    celula *p;
    for (p = listaEncadeada; p != NULL; p = p->prox)
        printf ("%d\n", p->conteudo);
}
```

Listas Simplesmente Encadeadas

Inserir no FINAL:

Função INSERIR colocando os elementos sempre no final da lista

```
celula *inserir(int x, celula *p){
    celula *novo, *topo;
    topo = p;
    novo = malloc (sizeof (celula));
    novo->conteudo = x;
    novo->prox = NULL;
    if (p!=NULL){
        for(p;p->prox!= NULL;p=p->prox);
        p->prox=novo;
    }else topo = novo;
    return topo;
}
```

Referências

Referências


CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262032937.